Integrate WSN to the Web of Things by Using XMPP

Pin Nie and Jukka K. Nurminen

Aalto University, School of Science, Finland
{pin.nie,jukka.k.nurminen}@aalto.fi

Abstract. Wireless Sensor Network is a promising technology thanks to its numerous beneficial applications. The recent trend towards Web of Things leverages substantial web technologies and toolkits, which greatly simplify the chore of WSN application development. However, the complex web server and heavy HTTP communications impose difficulties on portability of WSN applications and node's resources management. In order to provide a lightweight web integration and uniform data representation, we propose to employ XMPP, an open standard formalized by IETF, to build instant messaging and presence service for wireless sensor nodes. In this paper, we develop a scalable and flexible XMPP sensor bot to integrate WSN into generic XMPP architecture. We also design two lightweight XMPP extensions for sensor node representation and task configuration. The efficient XML expression in our extension protocol can squeeze the payload into a single IEEE 802.15.4 packet and does not cause XMPP message fragmentation. Our solution works directly on MAC layer without the need of TCP/IP stack. Based on our sensor bot, we propose a novel application for product validation and customer behavior analysis with RFID/NFC technology on smartphones to demonstrate a new context-aware service.

Keywords: Wireless Sensor Network, XMPP, Instant Messaging and Presence Service, RFID/NFC Application.

1 Introduction

Humans have intrinsic limitations to observe the surrounding world in terms of the sensitivity of our natural perception of the environment, the variety of our senses and the persistent working capability. Moreover, we cannot directly perceive hazardous substances, such as chemicals and radiation. Therefore, we need sensors to enhance our perception, to perform persistent monitoring and to expand our awareness. Particularly, wireless sensor networks (WSN) [1] connect different sensor nodes over distance to fulfill various tasks in a cooperative manner without costly cable infrastructure.

Despite many benefits, WSN has not yet been widely employed in practice for multiple reasons. Firstly, there are numerous wireless technologies, such as IEEE 802.15.4/Zigbee, WirelessHART, Bluetooth Low Energy (BLE), Near

F. Martins, L. Lopes, and H. Paulino (Eds.): S-Cube 2012, LNICST 102, pp. 105–120, 2012.

Field Communication (NFC), Z-wave, Dash7 and WiFi. These protocols and standards do not work together and cause severe interoperability issue when bridging multiple WSN applications. Secondly, resource constraints on sensor nodes set up physical barriers to build reliable and long lifespan WSN systems. Thirdly, incompatible proprietary WSN software create information silos and hinder application development. Consequently, these difficulties prohibit cooperation between different WSN applications for seamless service creation. To solve this problem, web integration brings a mature platform for better flexibility and scalability. Evolved from the Internet of Things (IoT) [2], the Web of Things (WoT) [3] is proposed to simplify WSN development by leveraging embedded web server and HTTP communications. This architecture fits a few regular WSN applications with powerful sensor nodes. However, the complex web server and heavy HTTP communications impose difficulties on portability of WSN applications and node's resources management. Many limited sensor motes need ultra-lightweight web integration in just a few kilobytes memory [4]. Furthermore, HTTP does not support real-time communication which is required in safety-critical and/or highly interactive WSN applications.

eXtensible Messaging and Presence Protocol (XMPP) [5] [6] is an open standard, XML-based communication protocol for instant messaging and presence information. XMPP can offer uniform data representation and real-time service. XMPP has been tested in some WSN application scenarios [7] [8] to disseminate data. Therefore, we propose to extend XMPP architecture for WSN integration on the Web. In this paper, we develop an XMPP client called *sensor bot* to integrate sensor nodes into generic XMPP architecture. In contrast to the static customized messages sent by other XMPP/Jabber bots, our sensor bot automatically generates events derived from sensor data based on the processing rules, and then encapsulates events into XMPP messages for further distribution. In addition, the sensor bot also accepts incoming requests in predefined patterns from authorized contacts to create new tasks and/or update existing tasks. Moreover, we add feedback study on user rating from their responses for parameters optimization, such as sampling frequency and/or event notification preference. Essentially, we explore three generic programming components to build a scalable and flexible XMPP bot for more WSN application logics and better interaction with end users. Meanwhile, to improve XMPP functionality regarding WSN characteristics, we design two XMPP extensions for sensor node representation and task configuration. Since many tiny wireless sensor nodes do not support TCP/IP stack, we build our XMPP extensions directly on MAC lay and squeeze the payload into a single IEEE 802.15.4 packet. Hence, our protocol completely removes the dependence on transport layer by eliminating XML message fragmentation and data packet serialization.

Based on our WSN-enabled XMPP architecture, we propose a novel application for product validation and customer behavior analysis. We use RFID/NFC technology on smartphones to build a new context-aware service. We measured end-to-end latency to evaluate real-time performance on a preliminary prototype. The experimental result proves the feasibility to integrate WSN to the

Web via open XMPP network on the Internet. Thus, our XMPP architecture can be reused by other WSN applications.

The rest of the paper is organized as follows: Section 2 studies related works on WSN web integration and compares with our XMPP-based solution. Section 3 explains the primary elements in XMPP network and our development of sensor bot. Section 4 presents two XMPP extensions for WSN integration. Section 5 elaborates a novel application on smartphones with RFID/NFC technology. At last, Section 6 summarizes the paper.

2 Related Works

The Web of Things is an emerging architecture with the purpose to connect a variety of limited devices over the Internet, such as mobile phones and wireless sensor nodes. In [9] [10], a web mashup is proposed for embedded devices using tiny web server through either a gateway or direct integration on the SunSPOT [11] sensor node with a built-in HTTP engine. To expand data sharing, [12] presents a WSN integration into social networks via Restful Web API. An extensive study on the integration of sensors and social networks is provided in [13]. A heavier and more sophisticated framework is Sensor Web Enablement (SWE) [14]. Open Geospatial Consortium (OGC) [15] has developed comprehensive SWE standards to integrate sensor devices, measurements, information models and services into the Internet.

Inspired by these novel development, we identify two critical elements to boost WSN integration to the Internet. The first element is a uniform data representation to encapsulate information in a widely supported format. The second element is a ubiquitous service to distribute information in a widely supported mechanism. Driven by these two considerations, we select XMPP as the platform to develop WSN web integration. XMPP employs universal XML data model to encapsulate general content into message payload for instant messaging and presence service, which is a generic data exchange mechanism widely supported by many web services. As an open industrial standard, XMPP has fostered a big user community [1] and numerous interoperable free software [2].

Compared with HTTP-based web server architecture, XMPP has two advantages for WSN applications. Firstly, XMPP is a real-time protocol which can better serve safety-critical WSN applications, e.g., fire detection and health care. Secondly, XMPP client is a more lightweight program than a web server in typical Client/Server (C/S) model in which most application logics reside at server side. Thus, XMPP client can fit into embedded systems and mobile devices easier. Although XMPP has developed a lot of extensions, the core feature is rather straight-forward and simple with three basic XML stanzas. In contrast to static webpages and *pull* method in HTTP, XMPP supports dynamic data transportation with *push* method which is more suitable for asynchronous event

[1] Jabber Organization: `http://www.jabber.org/`
[2] XMPP Standards Foundation: `http://xmpp.org/`

notification in WSN applications. Concerning XML efficiency on resource consumption, several binary XML protocols are being standardized, such as Efficient XML Interchange (EXI) [16] [17] formalized by W3C. Binary XML compresses verbose XML content into more efficient format for storage and transmission. Notice that existing EXI products [18] [19] can achieve two orders of magnitude smaller binary XML messages with less than 2MB implementation.

XMPP/Jabber bot [20] is a client program sending customized replies in response to time trigger and/or incoming messages from others. Following publish/subscriber paradigm [21] [22], XMPP bot provides events notification service to other peers in the contact list. This feature saves considerable time and energy for WSN data dissemination from blind polling new data at a fixed interval. One micro-XMPP implementation [23] demonstrated the feasibility to enable XMPP on resource constrained wireless sensor nodes. In comparison, our XMPP sensor bot excels in flexibility of data processing and payload efficiency with two lightweight XMPP extensions tailored for WSN packet.

3 XMPP Architecture for Wireless Sensor Networks

Due to the limited battery and long lifetime requirement, WSN is featured with narrowband communication with low processing capabilities. In contrast to the fast hardware upgrade in many embedded systems, wireless sensor nodes are scaling up slowly for two reasons. Firstly, most of WSN applications mainly require primitive sensory data to enable situation awareness. WSN serves as an interface between cyberspace and physical world. Sensing is the fundamental objective rather than processing. Secondly, current battery technology cannot power small sensor nodes for long lifespan and rich processing at low cost, particularly in the applications where large data is sampled (e.g., vibration, image and acoustic signals). Hence, we choose three types of devices, namely smartphone, sink node and powerful sensor nodes as the gateway to host XMPP sensor bot. In this section, we firstly introduce the generic XMPP framework and then explain our development of WSN-connected XMPP sensor bot.

3.1 Generic XMPP Framework

The XMPP architecture as shown in figure 1 is composed of three components: XMPP client, XMPP server and gateway to foreign networks. The XMPP client is an I/O interface for text and multimedia rendering and sending. The XMPP server is responsible for connection management and message routing. The gateway bridges different networks by translating different protocols into XMPP and vice versa. Any two XMPP elements use TCP connection for XML streaming session. A single TCP connection can carry multiple sessions identified by their unique ids. The identifier for XMPP entity (a.k.a., JID) follows a URI pattern: *user@domain/resource*. Three basic XML stanzas are defined in XMPP as follows. XML stanza is the message payload exchanged over the XML stream.

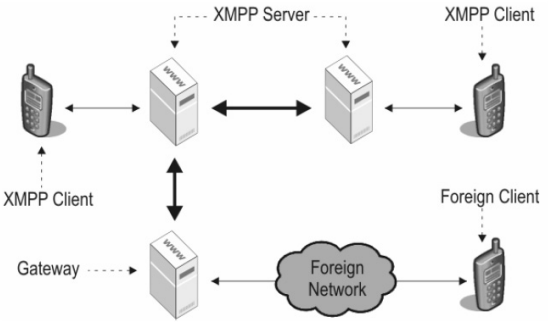

Fig. 1. The XMPP Architecture [24]

– <message/>: it is unicast carried out in store-and-forward mechanism through which one entity pushes information to another transferring messages between two endpoints.

```
<message from='niepin@aalto.fi' to='jukka@hut.fi'
         type='chat'>
  <body>Hello</body>
</message>
```

– <presence/>: it is broadcast executed in publish-subscribe mechanism through which multiple entities receive information about an entity to which they have subscribed, i.e., entity's availability.

```
<presence from='niepin@aalto.fi' xml:lang='en'>
  <show>online</show>
  <status>Working in the office</status>
</presence>
```

– <iq/> (Info/Query): it is a ping-pong interaction between two entities in request-response mechanism. This stanza can be used for service discovery and resource retrieval, such as file transfer and roster fetch. IQ interactions follow a common pattern of structured data exchange in the type of either get/result or set/result. An unique id is required to identify a transaction.

```
<iq type='get' from='niepin@aalto.fi' id='vq71f4nb'>
  <query xmlns='jabber:roster'/>
</iq>
```

XML stanzas must reside in the <stream/> block, which stands for an XML stream. All stream-level errors are unrecoverable and an <error/> stanza with description is sent by the detecting entity. Security constructs are specified in XMPP core protocol [5]. Two security protocols are employed to provide confidentiality, data integrity and entity authentication. The first protocol, Transport Layer Security (TLS), lies on the top of TCP connection and encrypts

XML streams between two entities. TLS protects XMPP channel from tampering and eavesdropping. On the top of TLS is the second protocol, Simple Authentication and Security Layer protocol (SASL), which provides a reliable mechanism to validate the identity of an entity. Prior to the SASL negotiation, XMPP clients should use TLS to secure the XML streams with "STARTTLS" extension in the namespace "urn:ietf:params:xml:ns:xmpp-tls". SASL defines a generic method for adding authentication support to connection-based protocols in the namespace "urn:ietf:par-ams:xml:ns:xmpp-sasl". Supported security mechanisms are announced within a <mechanisms/> element for negotiation between two XMPP entities. In addition to TLS and SASL, another XMPP specification [25] provides end-to-end (E2E) signing and object encryption.

Among plentiful XMPP extensions, Multi-User Chat (MUC) [26] is an important specification to enable many-to-many communication. This feature has high value to interconnect different WSN applications and share data for context-aware information processing. MUC is also beneficial to link up multiple WSNs at different locations for a bigger scale WSN application of the same interest or subject. All we need to do is to create a chat room, give a certain subject and invite others to join. A participant could be a WSN-enabled XMPP client publishing data and human users observing events at runtime.

3.2 XMPP Sensor Bot

From the generic XMPP framework above, we can leverage three key features for WSN applications as below:

1. Interactive communication: XMPP uses <presence/> stanza to indicate availability of an entity and <message/> stanza to exchange text messages in both online and offline cases. The presence status suits duty-cycled WSN applications in which sensor nodes are scheduled to sleep periodically for energy preservation. Whenever the node(s) wakes up and transmits data, associated contacts in the group will receive its updated information. Meanwhile, users can send queries or commands to get latest data or reconfigure tasks if authorized. When sensor nodes return to sleep, all offline messages will be cached in XMPP server till the next awake period.
2. Service discovery: based on the <iq/> stanza, an XMPP extension, XEP-0030 [27] specifies service discovery process for XMPP entities. We can define a JID URI pattern "wsn_name@domain/sensor_node_id/sensor" for resource binding on a specific WSN within the given domain. This URI allows one WSN operator to have multiple WSN applications which are identified by their names. Each WSN application may consist of many nodes and each node may equip several sensors for different measurements. The hierarchical structure provides flexible and scalable WSN resource binding.
3. Group chat: XMPP extension MUC offers rendezvous point for multiple WSN operators to share their real-time data in a common interest. This feature not only creates context-rich WSN application, it also covers a wide physical area. Based on a list of MUC room subjects, users can choose their favorite WSN application and observe a specific environment or monitor an interesting object.

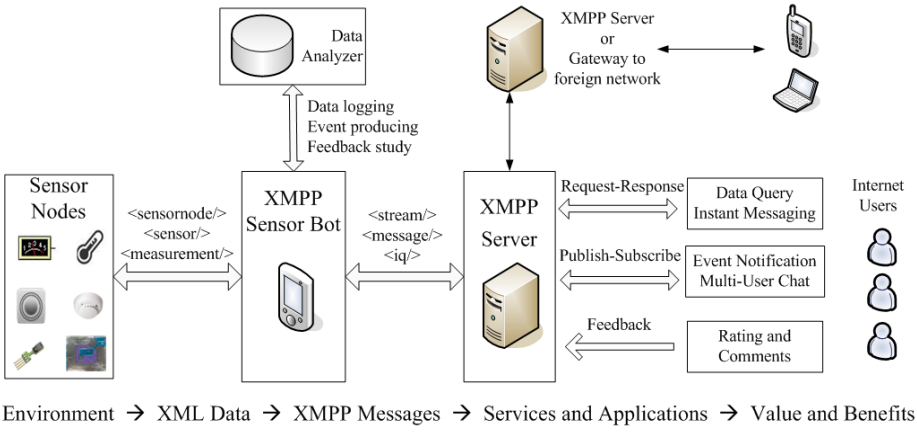

Fig. 2. WSN-enabled XMPP Architecture

In order to integrate WSN into generic XMPP architecture with minimal cost, we develop an XMPP sensor bot. Figure 2 illustrates the overall system from the sensor node to the Internet user. We create a small XMPP client [3] to collect data from sensor nodes and send customized messages to remote XMPP entities. A data analyzer is built into the sensor bot for data logging, event producing and feedback study. All other XMPP entities remain the intact without any change. Information flows smoothly from physical world to the Web via XMPP networks. Thus, our solution simplifies WSN application and service creation.

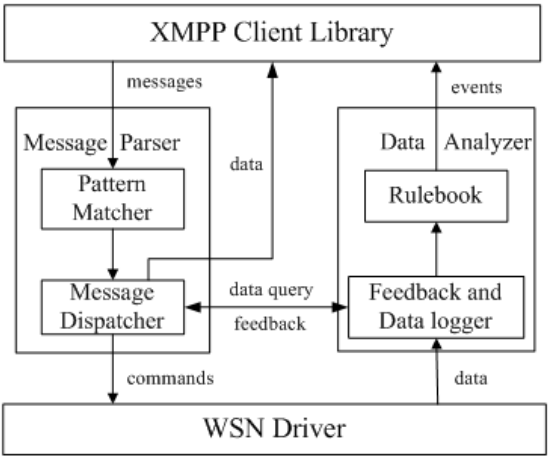

Fig. 3. Program structure of the sensor bot

[3] Smack API: `http://www.igniterealtime.org/projects/smack/`

To realize a seamless connection between WSN and XMPP network, we explore three programming components in the sensor bot. Figure 3 illustrates the program structure and information flows. Firstly, we add the WSN driver provided by the manufacturer into the XMPP client to capture packets from the serial port which listens to wireless sensor nodes. The driver also handles network management, such as node leaving and joining. A typical setup is to employ master-slave mode in star topology for single-hop WSN. In this case, one sensor bot can manage several wireless sensor nodes in a synchronized duty-cycled manner. For a large WSN with multi-hop connections, the WSN driver should work separately to guarantee fast process of the large amount of incoming packets. The sensor bot reads data from local file system or database periodically based on a timer. Secondly, we develop a *rulebook* to specify WSN application logics for data processing. The rulebook is an XML file defining filtering conditions and publishing events for each type of sensor. Triggered by new receiving packets from the WSN or a local timer, the data analyzer executes these rules to publish events to the subscribed peers in the contact list through XMPP <message/> stanza in multicast method. The following example shows two types of sensor with different filters and consequent events. The first temperature sensor applies threshold filter and generates alarm if the value exceeds 40 degrees. The second accelerometer sensor applies deviation filter on vibration measurement and detects impact during motion monitoring. The rulebook is flexible and scalable to contain more data aggregation techniques [28] [29].

```
<sensor type='temperature'>
  <filter id='threshold' operator='>' value='40'/>
  <event id='alarm' description='temperature is too high.'/>
</sensor>

<sensor type='accelerometer'>
  <filter id='deviation' operator='>' value='1'/>
  <event id='motion' description='impact is detected.'/>
</sensor>
```

The third programming component is a message parser. The parser reads incoming data queries, tasks configuration commands and feedback ratings. To support flexible interactions with users, we define a few patterns using regular expression to differentiate diverse requests. Our parser extracts sensor type, command parameters and filtering conditions from the received message. Accordingly, the sensor bot either returns the latest measurement for data query or set command parameters (e.g., LEDs blink in a specific color and order). If an incoming message does not comply with any expression patterns, a list of allowable patterns will be replied automatically to the requester. Unlike event multicast defined in the rulebook, data query is unicast and executed only once in request-response mechanism. On the one hand, the event notification is used in routine monitoring to avoid overwhelming raw data and to highlight special status and/or changes. On the other hand, real-time data query handles random and dynamic situations, such as customer service.

Tasks configuration and parameters optimization are important issues in WSN application development. Thus, we add feedback study to rate users satisfaction. Our message parser will prompt users to "like" or "dislike" the response. Subsequent comments will be recorded as reasons for feedback study later. By counting the number of "like" and "dislike", the WSN operator can evaluate the popularity and quality of his applications and services for further improvement. Moreover, our sensor bot supports an open control feedback loop based on the user's presence information. By reading the subscriber's presence, the sensor bot decides whether it should send the data right now or postpone when the remote user is busy or not available.

4 XMPP Extensions for Sensor Networks

In the program structure of our XMPP sensor bot above, WSN driver is a potential bottleneck, because sensor nodes do not use XML format to encapsulate their data. The sensor bot has to parse every packet and format into XML element for XMPP messages. As a result, packet transformation may exhaust the sensor bot when dealing with heavy network traffic. Furthermore, low level packet conversion also hinders XML parser development when porting a WSN application to another XMPP sensor bot. However, none of existing XMPP extensions suit WSN due to their huge resource consumption in complex signaling and verbose expression. To solve this problem, we propose two lightweight XMPP extensions to encapsulate XML format payload into a single IEEE 802.15.4 packet. Our XMPP extensions cover two fundamental functions in all WSN applications, namely node representation and task configuration.

4.1 Node Representation

There are two common attributes for every sensor node: capability and measurement. Capability specifies the equipped hardware sensors and the supported precision and format. One sensor node may have several sensors (e.g., temperature, light, accelerometer, barometer) onboard. Measurement delivers data from the hardware. One temperature sensor gives float value in degree centigrade. Measurement may also include data point of embedded software algorithms. One sensor node can calculate dew point based on temperature and humidity at a given altitude. With these two common attributes, a WSN application supports service discovery and data provision.

To represent a sensor node in a concise profile, we design a new XML stanza <sensornode/> with two child elements: <sensor/> and <measurement/>. As aforementioned, these two elements list equipped sensors and supported measurements onboard. In addition, a <sensornode/> has one basic attribute, 'id' and three optional attributes, 'type', 'location' and 'time'. The 'id' attribute identifies the node in WSN and can be appended in XMPP entity URI for resource binding. The 'type' attribute is a tag for grouping or classification depending on the WSN application. The 'location' attribute gives positional context information. The 'time' attribute gives measurement a time stamp and may

also be used to update node's presence status. The following example shows two <sensornode/> stanzas. The first stanza is for service discovery at the initial stage and the second stanza is a regular data report.

```
<sensornode id='node_1' type='fire_detector'
            location='office' time='YY:MM:DD-HH:MM:SS'>
  <sensor type='temperature' unit='celsius'/>
  <sensor type='light' unit='lux'/>
</sensornode>

<sensornode id='node_1'>
  <measurement type='temperature' value='25'/>
  <measurement type='light' value='100'/>
</sensornode>
```

Once the initial stage is completed, we reduce syntax verbosity for subsequent measurement reports by removing quotation marks and using abbreviated letter(s) of every XML element and attribute, such as 'sn' for 'sensornode' and 'y' for 'type' as shown below. We also encode measurement types in one byte which provides 255 unique options, enough for all possible values. Assume node's id and every letter take one byte and every numeric value takes four bytes (floating point), the abbreviated expression in the following example saves up to 64% space compared with the previous version. Assume employing the IEEE 802.15.4 radio standard which allows maximum 102 bytes for payload per frame, a single packet can take up to 5 measurements including location and time attributes in a row. We use a flag 'wsn=true' in the sensor bot to indicate this compact payload strategy in XML format for efficient wireless transmission. In the case of big chunk of data set, such as vibration amplitudes and acoustic signals, only the first and the last packet use XML format to mark the start and the end of packet streaming. The rest packets in the middle contain only values for serialization.

```
<sn i=node_1>
  <m y=t v=25/>
  <m y=l v=100/>
</sn>
```

4.2 Task Configuration

The second extension uses 'get/set' methods in XMPP <iq/> stanza to execute task configuration. We use get/result transaction to fetch configurable parameters on a sensor node and set/result transaction to update parameters' value. Since all parameters belong to either hardware sensors or software algorithms, we embed a new XML element <param/> into <sensor/> and <measurement/> elements. In the following example, a sensor node returns sampling frequency for accelerometer sensor and two embedded calculations for light measurement, average value and threshold filter.

```
<iq type='get' from='niepin@aalto.fi'
    to='node_1' id='info_1'>
  <query xmlns='http://aalto.fi/wsn#parameters'/>
</iq>

<iq type='result' from='node_1'
    to='niepin@aalto.fi' id='info_1'>
  <query xmlns='http://aalto.fi/wsn#parameters'>
    <sensor type='accelerometer'>
        <param name='frequency' unit='Hz'/>
    </sensor>
    <measurement type='light'>
        <param name='average'/>
        <param name='threshold'/>
    </measurement>
  </query>
</iq>
```

When setting parameters, our extension allows independent use of <param/> to activate or deactivate a sensor or an embedded algorithm. In the following example, we deactivate accelerometer and set up a threshold filter for light sensor. Successful result or error response is replied from the sensor node. Note that error message uses predefined code to indicate possible reason.

```
<iq type='set' from='niepin@aalto.fi'
    to='node_1' id='config_1'>
  <query xmlns='http://aalto.fi/wsn#parameters'>
    <param name='accelerometer' value='false'/>
    <measurement type='light'>
        <param name='threshold' operator='>' value='100'/>
    </measurement>
  </query>
</iq>
```

If success case

```
<iq type='result' from='node_1'
    to='niepin@aalto.fi' id='config_1'/>
```

If error case (404 parameter not found)

```
<iq type='error' code='404' from='node_1'
    to='niepin@aalto.fi' id='config_1'/>
```

Like the previous extension for node representation, the extension for task configuration also supports abbreviated expression. Given a certain namespace in a moderate size of WSN (nodes number< 255), we assume all id and type values in XML elements can be encoded in one byte letters and numeric values take four bytes (floating point). Then, the set/result example above can be squeezed into a single IEEE 802.15.4 radio packet as below:

```
<iq y=s f=niepin@aalto.fi t=node_1 i=config_1>
  <q x=http://aalto.fi/wsn#parameters>
    <p n=a v=f/>
    <m y=l>
        <p n=t o=> v=100/>
    </m>
  </q>
</iq>
```

If success case

```
<iq y=result i=config_1 f=node_1 t=niepin@aalto.fi/>
```

If error case (404 parameter not found)

```
<iq y=e c=404 i=config_1 f=node_1 t=niepin@aalto.fi/>
```

Our XMPP extensions introduce a few new XML elements tailored for WSN and reuse XMPP core specification. The abbreviated expression presents an efficient XML format for wireless transmission. In a typical WSN standard IEEE 802.15.4/Zigbee [4], no packet fragmentation is needed. Therefore, our solution achieves good scalability and interoperability by applying uniform XML format at little cost of payload redundancy.

5 RFID/NFC Application for Product Validation and Customer Behavior Analysis

Fake products and expired food are two big problems in many countries. How to validate products or food in a short time is a challenge for three reasons. Firstly, there are numerous products and food in the world or just in a local grocery store. Centralized method is impossible to accommodate all products and to keep up-to-date information. Secondly, product validation service has to be ubiquitous for easy access. Meanwhile, this service should be low cost to motivate people for daily use. Thirdly, the whole validation process must be quick and responsive to massive requests.

Driven by three concerns above, we propose to use smartphones and our XMPP sensor bot to build a fast product validation service with RFID/NFC technology. Correspondingly, there are three advantages in our solution. Firstly, XMPP provides an open peer-to-peer architecture for direct and instant communication between producers and consumers through instant messaging and presence service. Secondly, smartphone is becoming a ubiquitous mobile computing device in people's daily life. High speed wireless technologies, such as Wifi and 3G, support wide connectivity via smartphones. Furthermore, equipped with GPS receiver, smartphones can provide location and time context information for accurate data processing. Thirdly, RFID/NFC technology on smartphones

[4] Zigbee Alliance: `http://www.zigbee.org/`

enables touch-and-see quick object identification. RFID tag is a cheap passive circuit which can be attached on any solid product.

We develop an XMPP sensor bot on a smartphone which equips NFC chip to read RFID tag on the product. RFID tag should contain at least two types of data, the product's series number and the JID of the producer's customer service. Meanwhile, the sensor bot also reads current location (e.g., GPS coordinate) and the time. By encapsulating all these information in an XMPP <iq/> stanza, the sensor bot sends a product validation request to the remote producer and displays the result on the smartphone. In this way, the customer can easily validate a product or food at real-time without typing or searching anything.

Similar to the barcode used in warehouse products management, the validation request consists of four key elements: <product ID, timestamp, location, dealer's signature>. By checking the time and the location attribute, the producer can validate if the product/food has expired or not. Fake products with invalid IDs or obsolete valid IDs will not pass the validation because a product's series number is unique in a specific area during a certain period. The dealer's signature provides basic security feature with a shared secret which can be used to hash the whole validation request. The producer may also add promotion information in the response for advertisement. In the following example, the producer confirms the valid product and also replies discount information. The dealer's signature is omitted for simplicity.

```
<iq type='get' id='product_series_number'
    time='YY:MM:DD-HH:MM:SS' location='City_District_X'
    from='customer@example.com' to='producer@company.com'>
  <query xmlns='product#info'/>
</iq>

<iq type='result' id='product_series_number'
    from='producer@company.com' to='customer@example.com'>
  <query xmlns='product#info'>
    <item name='product_name' type='wine' value='true'/>
    <item name='promotion' type='discount' value='promo_code'/>
  </query>
</iq>
```

We implemented a preliminary testbed to evaluate the feasibility and real-time performance. Our testbed uses the smartphone Nokia C7 to read RFID tag and forward the data over 3G/WiFi to a proxy which runs a script to encapsulate incoming data in XMPP messages and then send to the Gtalk client on another smartphone and a Jabber client on a laptop. Figure 4 illustrates the network structure of our testbed. We measured the end-to-end latency. It takes about three seconds from touching the RFID tag with one smartphone till the XMPP message appearing on the other smartphone and the laptop. A round-trip time (RTT) may take six seconds. The major latency comes from the middle box, a HTTP-XMPP proxy which performs protocols translation for two-way communication. Our next step is to remove this extra proxy and implement

peer-to-peer XMPP messaging between the smartphone and a Jabber client. The native XMPP communication will be much faster without any middle box. In addition, we also notice that 3G connection adds longer delay than WiFi connection due to the extra signaling overhead with the base station.

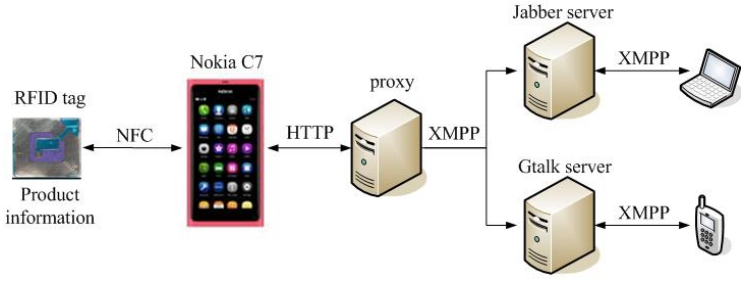

Fig. 4. Product validation with smartphone based on XMPP

In order to handle large number of request at low cost, the producer can register a JID on public XMPP servers and modify an open-source XMPP client [30] to automate product validation with a backend product database. Moreover, the producer can use this service to study customer behavior by counting the number of requests and exploring context information. This application can provide insights to three important questions that many producers concern:

1. Popularity: what is the total number of received requests during a certain period?
2. Customer distribution: at what time and in which place do customers buy this product?
3. Genuine-to-counterfeit ratio: how many fake products exist in the market?

6 Discussion and Conclusion

On highly constrained sensor nodes which cannot afford resources for TCP/IP stack, our lightweight XMPP extensions work directly on MAC layer for efficient XML encapsulation and transmission. More powerful devices can host our XMPP sensor bot, such as a smartphones, sink node connected with laptop or PC and high-end wireless sensor nodes (e.g., SunSPOT and Imote2-linux). Recently, IPv6 is gaining increasing adoption by many embedded devices and sensor nodes [31]. A number of efficient and reliable transport protocols have been also proposed on the lower layers for WSN [32]. Thus, our XMPP sensor bot will likely be able to run on low-end wireless sensor nodes in the future. Consequently, machine-to-machine communication will connect multiple XMPP sensor bots together to create heterogenous WSN applications at larger scale.

In this paper, we propose to use XMPP to integrate WSN to the Web of Things. Compared with another architecture based on web server and HTTP

communications, our XMPP sensor bot can achieve real-time performance with small program footprint. We design two XMPP extensions for sensor node representation and task configuration. By reusing existing XMPP standards, our solution makes it easier to integrate WSN on generic XMPP architecture. At last, we propose a novel application for product validation with RFID/NFC technology to demonstrate the feasibility of our solution.

Acknowledgements. This research is partly funded by TEKES in the Internet of Things programme of TIVIT (Finnish Strategic Centre for Science, Technology and Innovation in the field of ICT).

References

1. Kuorilehto, M., Hännikäinen, M., Hämäläinen, T.D.: A survey of application distribution in wireless sensor networks. EURASIP Journal on Wireless Communications and Networking 2005, 774–788 (2005)
2. Atzori, L., Iera, A., Morabito, G.: The internet of things: A survey. Computer Networks 54 (October 2010)
3. Guinard, D., Trifa, V., Mattern, F., Wilde, E.: From the Internet of Things to the Web of Things: Resource Oriented Architecture and Best Practices. Springer (2011)
4. Mottola, L., Picco, G.P.: Programming wireless sensor networks: Fundamental concepts and state of the art. ACM Computer Survey 43 (April 2011)
5. Saint-Andre, P.: RFC 6120 Extensible Messaging and Presence Protocol (XMPP): Core (2011)
6. Saint-Andre, P.: RFC 6121 Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence (2011)
7. Goncalves, J., Ferreira, L.L., Chen, J., Pacheco, F.: Real-Time Data Dissemination for Wireless Sensor Networks using XMPP. Polytechnic Institute of Porto, Tech. Rep. (2009)
8. Hornsby, A., Belimpasakis, P., Defee, I.: XMPP-based wireless sensor network and its integration into the extended home environment. In: IEEE 13th International Symposium on Consumer Electronics, ISCE (2009)
9. Guinard, D., Trifa, V.: Towards the web of things: Web mashups for embedded devices. In: International World Wide Web Conference, Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web, MEM 2009 (2009)
10. Guinard, D., Trifa, V., Wilde, E.: A resource oriented architecture for the web of things. In: Internet of Things, IOT (2010)
11. Sun small programmable object technology (sun spot) theory of operation, Tech. Rep. (2007)
12. Guinard, D., Fischer, M., Trifa, V.: Sharing using social networks in a composable web of things. In: 2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops, PERCOM (2010)
13. Aggarwal, C.C., Abdelzaher, T.: Integrating sensors and social networks. In: Social Network Data Analytics, pp. 379–412. Springer, Heidelberg (2011)
14. Bröring, A., Echterhoff, J., Jirka, S., Simonis, I., Everding, T., Stasch, C., Liang, S., Lemmens, R.: New Generation Sensor Web Enablement. Sensors 11 (2011)

15. Botts, M., Percivall, G., Reed, C., Davidson, J.: OGC Sensor Web Enablement: Overview and High Level Architecture (2007)
16. Schneider, J., Kamiya, T.: Efficient XML Interchange (EXI) Format 1.0 (2011)
17. Cokus, M., Vogelheim, D.: Efficient XML Interchange (EXI) Best Practices (2007)
18. Peintner, D.: EXIficient: an open source implementation of the W3C Efficient XML Interchange (EXI) format specification in Java (2011), http://exificient.sourceforge.net/
19. Inc., A.: Efficient XML (2011), http://www.agiledelta.com/product_efx.html
20. Google wave bots (2011), http://googlewavebots.info/wiki/
21. Albano, M., Chessa, S.: Publish/subscribe in wireless sensor networks based on data centric storage. In: Proceedings of the 1st International Workshop on Context-Aware Middleware and Services: Affiliated with the 4th International Conference on Communication System Software and Middleware, COMSWARE 2009 (2009)
22. Millard, P., Saint-Andre, P., Meijer, R.: XEP-0060: Publish-Subscribe (2010)
23. Hornsby, A., Bail, E.: μXMPP: Lightweight implementation for low power operating system Contiki. In: ICUMT 2009 International Conference on Ultra Modern Telecommunications and Workshops (2009)
24. Laukkanen, M.: Extensible Messaging and Presence Protocol (XMPP). University of Helsinki, Department of Computer Science, Tech. Rep. (2004)
25. Saint-Andre, P.: RFC 3923 End-to-End Signing and Object Encryption for the Extensible Messaging and Presence Protocol, XMPP (2004)
26. Saint-Andre, P.: XEP-0045: Multi-User Chat (2008)
27. Hildebrand, J., Millard, P., Eatmon, R., Saint-Andre, P.: XEP-0030: Service Discovery (2008)
28. Rajagopalan, R., Varshney, P.K.: Data aggregation techniques in sensor networks: A survey. IEEE Communications Surveys and Tutorials (2006)
29. Nakamura, E.F., Loureiro, A.A.F., Frery, A.C.: Information fusion for wireless sensor networks: Methods, models, and classifications. ACM Computing Surveys 39(3) (2007)
30. Use XMPP to create your own google talk client (2010), http://web.sarathlakshman.com/Articles/XMPP.pdf
31. Montenegro, G., Kushalnagar, N., Hui, J., Culler, D.: RFC 4944 Transmission of IPv6 Packets over IEEE 802.15.4 Networks (2007)
32. Ayadi, A.: Energy-efficient and reliable transport protocols for wireless sensor networks: State-of-art. Wireless Sensor Network (March 2011)